Dan Mopsick

Dr. Juan Arias

Software Development 1

May 8 2017

Mopsick CCD Project Final Write Up

Abstract

The goal of this project is to create a system that is capable of parsing XML Continuity of Care into Java objects and storing these objects in a database that can be accessed with a RESTful API. In order to parse a file, the filename must be passed to the system. The system can handle hard coded in files and can take in a file name from the user and store its contents into the database. The backend system is also connected to a front end system that allows users to view, create, and search for patients through HTML forms passed via PHP. The system also has the basic CRUD functions of being able to create, read, update, and delete patients.

Introduction

The motivation of this work is to create a easy to use and scalable system for transferring patient information from healthcare providers to the patients' themselves. The major issue of transferring patient information is patient confidentiality and security. This is an issue that extends farther than the scope of the project. The purpose of this project is to create the infrastructure for the passing of patient information and the establishing of protocols to process and read patient information from XML into Java. The RESTful aspects of the system are built using Spring Data and Spring Boot. The XML files are parsed utilizing javax's parsing package.

On top of the backend written in Java, a front end system was created to facilitate user interaction with the system via a browser in addition to using Postman. The front end of this project was built using HTML, JavaScript, CSS, and PHP. A front end system was created in addition to the back end in order to demonstrate the capabilities of the system. The front end also provides a more user friendly interface than the one provided in Postman. Using Postman requires one to have it installed, know how to use it, and to know the exact resting endpoints that they need to reach in order to test the project. Due to the focus of the class being on the backend Java programming, that is where the majority of the time was spent. The front end was designed with functionality in mind over aesthetics because it is meant to serve as an aid to demonstrate the back end of the project.

Detailed System Description

A user would use this system to receive patient information from the database or add in patients via an XML file that follows the protocol laid out in the sample files. This is be done by having the user use PostMan to make a POST request that passes the name without the file extension of the file that the user wishes to parse and add to the database. A user can access a specific patient by making a GET request with the id of the desired patient in the path. (ex: http://localhost:8080/patient/1). A user can return all of the patients saved into the database by making an empty GET request (http://localhost:8080/patient/). The class PatientController handles all of the requests from the user and uses PatientService and ParserService to interact with the PatientRepository and XMLDomParser respectively. This allows for loose coupling between the classes and classes that are built with one purpose in mind. The Patient Controller, Patient Service, and Parser Service class each have unit tests written for them to ensure that they function with intended behavior.

A user can also access limited functions of the system by using the front end system. The web server is currently hosted on my local machine. The web interface can be reached at http://localhost/pre-patient-info-client/src/pages/ when ran on my machine. Upon reaching this page the user is presented with a short welcome message and a navbar presenting four different options to showcase the ability of the system to store, access, and manipulate data. The four options available on the front end system are the following: add a patient, search for a patient with an Id, search for all patients with a specific family name, or view all patients in the database. The front end component of the system provides a simple to use interface, but does not allow the user to have as great of control over the system and limits the methods of the controller that can be accessed.

Accessing the system from Postman gives a more experienced user greater control over the system. Without Postman, the user is unable to delete a patient from the system, update a patient's info, or add patients into the system via an xml file. A more experienced user would be better off utilizing Postman due to the increased functionality. A less tech savvy or new user would should use the front end system due to its ease of access.

Requirements

The specific problem that this system is addressing is the lack of digital availability of a patient's medical records. Individual practices and healthcare providers currently have their own systems, but there is no standard system capable of efficiently handling patient information. This project seeks to build the infrastructure capable of storing patient information in a database and providing access to this database. The project seeks to build the infrastructure in such a way that it would be scalable to multiple healthcare providers and large amounts of patients. This project seeks to do this by creating a simple and consistent RESTful API. In addition to creating this web service, the project aims to present this data in an accessible and readable way with a front-end web system. The weakness with the current state of the system is that Spring Data does not naturally persist. The data only exists only while the server is running. In order to focus on the logic of the system, sample patients are created every time the project is launched in order to demonstrate the functionality of the system. This takes the stress of the project off of the database side of coding and onto the programming logic itself.

Literature Survey

Continuity of Care Documents were developed to provide a protocol for storing patient information in a standardized format within XML file. This is a step in the right direction. The XML format used in this project is based directly on the CCD format. For ease of access and in order to focus on the programming of this project, a simplified format was utilized. This project seeks to extend the protocol of CCDs to functional databases and RESTful web services that will provide patients and healthcare providers with access to this standardized information. Individual doctor's offices have their own similar patient information systems, but there is no standard in the healthcare industry. This project seeks to create the infrastructure for a patient information system that is simple enough to be used across several healthcare providers. This would solve the problem of incompatibilities amongst the information systems of different healthcare providers.

User Manual

A user has two ways to access this system. One method to access this system is through Postman as mentioned above. In this current state of the project, the system can accept two types of POST commands, two types of GET commands, DELETE commands, and POST commands. The user can create and add a patient into the database by passing a the arguments for a patient object in a JSON object. Another way to add patients into the database is by passing the name of an XML file located in the root directory to the system. (http://localhost:8080/patient/fileName). It is important to note that the file extension (.xml) should not be passed in the API call. The user can access a specific patient's information by making a GET call with the desired patient's Id. A user can also return every patient with a given family name with a GET request. The user can see the information of all the patients in the database by making a GET call with no path variable. The user can delete a patient from the database by making a DELETE request with the id of the patient that they desire to delete. The last type of request a user can make is a POST request. The user must specify the Id of the patient that they wish to edit and then pass a JSON object of a patient. These functions require knowledge of Postman and the system.

The other way to access the system is through the web based front-end. This usage method is designed to provide fewer, simpler actions to the user. Rather than passing a new patient via a JSON object, the front-end allows a user to create a new patient with an HTML form. This is much simpler to newer users and is closer to what the actual implementation would be in reality. No system is purely backend in today's web based society. A user is also able to view the information of a patient based on their Id in the database. Another GET request that the user can make simply is to return all patients with a given family name. The last function available on the web client is to display all of the patients in the database. To make the data more readable, the patients are presented to the user in sets of ten.

Conclusion

In conclusion, this project accomplishes it’s goal of creating a basic infrastructure for facilitating the transfer of patient information. The weaknesses of the system include validation, authorization, and authentication. The system functions fine given that all information is entered in the expected format, but improper input causes ugly errors. Currently security is a big issue for the system because the information of every single patient is available to every user. This could be improved by creating a user system to manage who is able to view what patients. The issues of scalability could be lessened by the addition of search functions. Searching by id becomes increasingly hard as the number of patients increases. This project does create a RESTful web service that provides a simple way to view and store patient information.

Reference/ Bibliography

Freeman, George, et al. "Continuity of care." Report of a scoping exercise for the National Co-ordinating Centre for NHS Service Delivery and Organisation R & D (NCCSDO). London: NCCSDO (2001).

Walker, Jan, et al. "The value of health care information exchange and interoperability." Health affairs 24 (2005): W5.